Claims cancelled:	none
Claims added:	19-27
Claims allowed:	none
Claims remaining in issue:	1-27

The Amendments

A typographical error in claims 3 and 12 has been corrected. New system claims 19-27 have been added, paralleling method claims 1-9 and computer program claims 10-18. No new matter has been added.

The Invention

The invention includes a system and method for parallelizing applications of certain script-driven software tools. In the preferred embodiment, scripts in the software tool scripting language are automatically analyzed in order to produce a specification for a parallel computation which is functionally equivalent to the original script. The parallel computation specification is then executed by a parallel runtime system, which causes multiple instances of the original software tool and/or supplemental programs to be run as parallel processes. The resulting processes will read input data and produce output data, performing the same computation as was specified by the original script. The combination of the analyzer, runtime system, original software tool, and supplemental programs will, for a given script and input data, produce the same output data as the original software tool alone, but has the capability of using multiple processors in parallel for substantial improvements in overall "throughput".

The §103 Rejection

The Examiner has rejected claims 1-18 under 35 U.S.C. §103(a) as being obvious over Brenner '149. Applicant respectfully traverses this rejection with respect to the claims in issue.

Brenner is not relevant art. Brenner teaches "a method of partitioning massively parallel processors, in particular those operating in a UNIX based environment, into a plurality of logical units while protecting the existing software installed on the system." Brenner '149, Abstract. The purpose is to:

> "... 'carve out' parts of the system that can run jobs without interfering with each other. ... Partitioning in general is the ability to divide up system resources into groups or parts in order to facilitate particular management functions. The

> structure of the MPP [massively parallel processing] system provides the opportunity to partition the system into groups of nodes for various purposes." col. 2, ll. 15-24.

The specific method of partitioning taught by Brenner is designed to protect existing software, applications, and other related data and information by preventing partitions from interfering with each other. In particular, a system unit [*i.e.*, the hardware environment] is partitioned into two or more sub-environments. In order to maintain these sub-environments isolated, all data is stored in a data repository located in the central control element. Data stored in this repository is then identified either as system data (data globally available to all nodes on all sub-environments) or as partitioned data (data accessible exclusively only to nodes residing in one of the partitioned sub-environments). When a querying node, also known as a client, requests data or information through the central control element, only data pertaining to the node's particular sub-environment is provided to it, in conjunction with other system or global data available to all nodes. A request by a node requesting data about a different sub-environment other than the node's respective sub-environment will be denied and not provided to that node.

Brenner teaches *nothing* about parallelizing a computer application program based on a script of a script-driven software tool. Indeed, a computer search of the text of Brenner indicates that the word "script" is used only once (col. 9, l. 1) in reference to how a boot script can access the System Data Repository (SDR) during recovery from a shutdown by using a second record containing the destination of a "persistent" system partition (col. 8, l. 52 over to col. 9, l. 4). Brenner does *no* analysis of *any* computer application program (let alone one based on a script) – Brenner simply creates sub-environments in the hardware system in which different programs can run without interference from each other.

Similarly, Brenner teaches *nothing* about *automatically* analyzing a script. Nor does Brenner teach anything about producing a *parallel computation specification* based on such analysis, where such parallel computation specification provides functional equivalence to the script when executed by a parallel runtime system.

The Examiner's specific arguments are also unsupported by Brenner. The Examiner cites to Brenner, col. 1, ll. 56-58, as teaching "a method for parallelizing a computer application program based on a script of a script-driven software tool." However, that section of Brenner

reads in its entirety as follows: "In addition, the users are often forced to install new levels of related software tools as well as their own applications on each node." This is a part of the description of the problem addressed by Brenner – that many MPP systems required that all nodes run on the same version of code, also using the same operating system and support programs, which made customization and workload management difficult, if not impossible. Using the same version of the code, the same operating system and same support programs could create migration concerns. Brenner states that the "sameness" requirement could make upgrading the system to a new level of code a potentially long and risky task, because users are often forced to upgrade each node to the new levels at the same time. Brenner, col. 1, ll. 51-56. This section has nothing whatsoever to do with "*parallelizing a computer application program based on a script of a script-driven software tool*" by "automatically analyzing the script and producing a parallel computation specification based on such analysis."

The other cites are equally inapposite – for example, the Examiner cites to col. 9, ll. 1-2 as teaching or suggesting use of SAS (a very specific script tool), yet Brenner only mentions a boot script used to recover from a shutdown. The Examiner cites to col. 8, ll. 44-50 as teaching or suggesting "at least one pre-defined parallelization rewrite rule [being] an algorithm selected from the group comprising simple partitioning, key-based partitioning, local-global division, external parallelism algorithm, and statement decomposition." The cite to Brenner is not even in the same universe of teaching – this section of Brenner relates to "using (IP) address to distinguish system partitions, [allowing] multiple daemons with the same name and function to be able to run on the same control workstation and using the same well-known port at the same time." Nothing about parallelization rewrite rule algorithms, let alone specific algorithms of the type claimed.

The same arguments apply to the remaining claims, many of which also distinguish from Brenner by the addition of additional limitations not taught or suggested by Brenner.

Conclusion

Accordingly, Applicant submits that none of the references, alone or in combination, anticipate or make obvious the invention as presently claimed. Applicant submits that this case is now in condition for allowance. Therefore, Applicant respectfully requests reconsideration and reexamination of the present application and allowance of the case at an early date.

Attached is a marked-up version of the changes being made by the current amendment. Applicant asks that all claims be allowed. Enclosed is one check for excess claim fee and the Petition for Extension of Time fee. Please apply any other charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: 4/12/01

Kenyon S. Jenckes
Reg. No. 41,873

PTO Customer No. 20985
Fish & Richardson P.C.
4350 La Jolla Village Drive, Suite 500
San Diego, CA 92122
Telephone: (858) 678-5070
Facsimile: (858) 678-5099

10090121.doc

Version with markings to show changes made

In the claims:

New claims 19-27 have been added.

Claims 3 and 12 have been amended as follows:

3. (AMENDED) The method of claims 1 or 2, wherein automatically analyzing the script includes:

(a) parsing the script into statements;

(b) constructing a serial dataflow graph from the parsed statements; and

(c) [construction] constructing a parallel dataflow graph from the serial dataflow graph.

12. (AMENDED) The computer program of claims 10 or 11, wherein automatically analyzing the script includes:

(a) parsing the script into statements;

(b) constructing a serial dataflow graph from the parsed statements; and

(c) [construction] constructing a parallel dataflow graph from the serial dataflow graph.